Project Three: Genetic Algorithm

Bradley Reeves
Sam Shissler

Department of Computer Science
Central Washington University
May 20, 2021

Contents

List of Figures

List of Tables

Introduction

The purpose of this project is to explore the genetic algorithm (GA) as a solution for two problems. The first problem is generating magic squares and the second is maximizing a function of two variables. These problems will be described in more detail later, but first, the GA will be described by its general application.

Like many concepts in computer science, the genetic algorithm draws its inspiration from biological processes. Specifically, it emulates biological evolution and natural selection. A high-level overview of this algorithm is given in Figure 1.1 below.

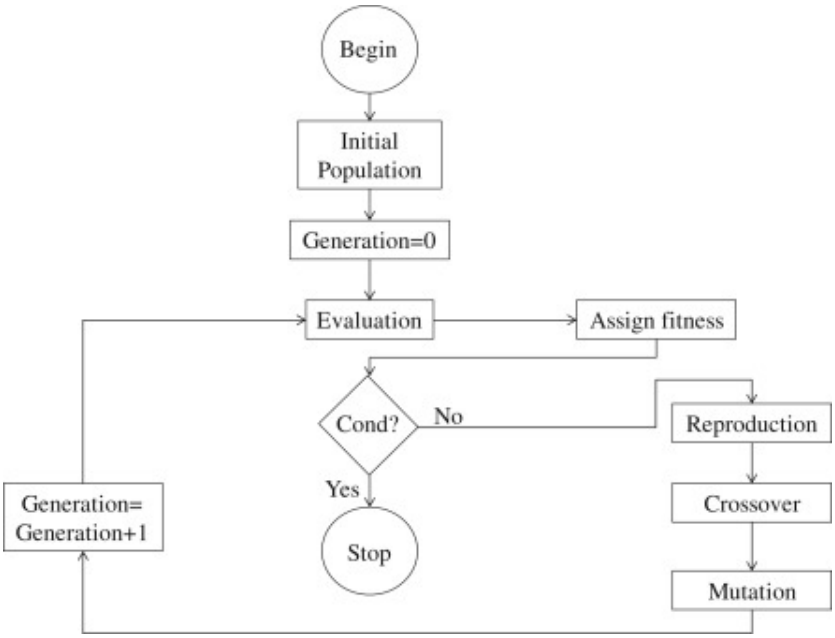

Figure 1.1: The Genetic Algorithm. [1]

To begin, an initial population is randomly generated. Next, each individual in the population is evaluated according to their "fitness". Fitness is a function

designed to measure how well the individual performs the given task. If the individual's fitness is optimal or meets some stopping criteria, the solution has been found. If no individual in the population meets the stopping criteria, then a new generation will be created. The new generation is created by first selecting two parents from the current population for breeding. There are several methods for doing this, but Fitness Proportional Selection (FPS) will be used throughout this project. FPS works by randomly selecting two individuals from the current generation according to some probability. Fitter individuals will have a higher probability of being selected. This ensures that fit individuals are selected most of the time, but there remains some diversity. The formula for calculating the probability of each individual being selected is given in Equation 1.1 where α is an individual and F^α is an individual's fitness. As shown, this probability is proportional to the individual's fitness.

$$p^\alpha = \frac{F^\alpha}{\sum_{\alpha\prime} F^{\alpha\prime}} \tag{1.1}$$

Note that this equation only works for problems where we are trying to maximize an individual's fitness. The higher the fitness, the higher the probability of being selected. For minimization problems, such as the magic square problem, simply substitute F^α with $\frac{1}{F^\alpha}$. For completeness, this formula is given in Equation 1.2.

$$p^\alpha = \frac{\frac{1}{F^\alpha}}{\sum_{\alpha\prime} \frac{1}{F^{\alpha\prime}}} \tag{1.2}$$

Once two individuals are selected for breeding, they become parents and some form of crossover takes place. In general terms, a crossover is an act of taking slices or parts from each parent and recombining those parts to form new individuals. Typically, two offspring will be generated from two parents. The primary forms of crossover include single-point crossover, multi-point crossover, and uniform crossover. These different forms are shown in Figure 1.2.

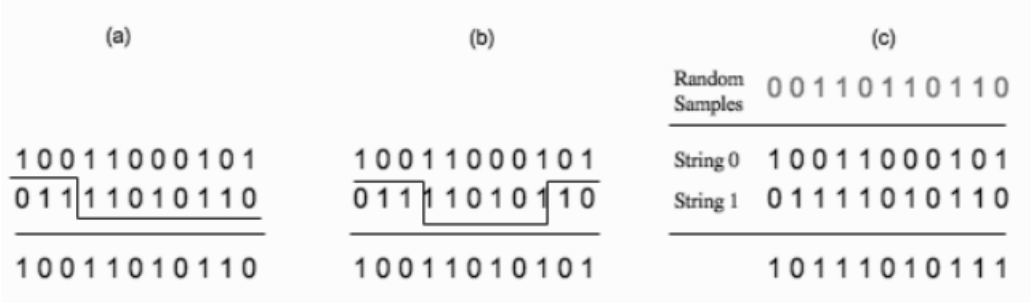

Figure 1.2: Crossover forms. (a) Single point crossover. (b) Multi-point crossover. (c) Uniform crossover. [2]

Single point crossover involves first selecting a random index. Next, each parent is "split" at that index. The first child is generated by combining the first part of the first parent and the second part of the second parent. Similarly, the second child is generated by combining the second part of the first parent with the first part of the second parent. Multi-point crossover follows the same logic, except that multiple random points are chosen for the permutation. Uniform crossover involves selecting a random element from the parents for each element of the child. Single point crossover will be used in the experiments presented in this paper.

After crossover is complete, the children may or may not undergo some mutation. Whether or not they are mutated depends on a probability p. A random float between 0 and 1 is generated and if this number is smaller than the defined mutation probability, the child is mutated. Mutation occurs by manipulating a single element within the individual. An example of this is shown in Figure 1.3.

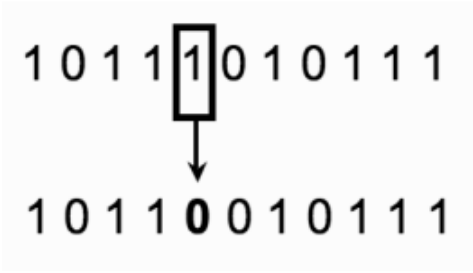

Figure 1.3: Mutating an individual. [2]

Finally, after the new generation is created, it replaces the current generation and starts the cycle over again. In this report, additional methods are also employed. One is the concept of elitism. Elitism involves taking the fittest n individuals in the current generation and transferring them into the new generation. The other method is tournaments. It is used in this project by selecting the two fittest individuals of the parents and new children. Whoever happens to be the fittest is put into the new generation. This entire process is repeated until an optimal value is found or the maximum number of generations has been reached.

Implementation

As mentioned previously, two separate problems are solved using the genetic algorithm in this project. The first problem involves generating a magic square. A magic square of order n is an arrangement of the numbers from 1 to n^2 in an n-by-n matrix, with each number occurring exactly once, so that each row, each column, and each main diagonal has the same sum. An example magic square is shown in Figure 2.1.

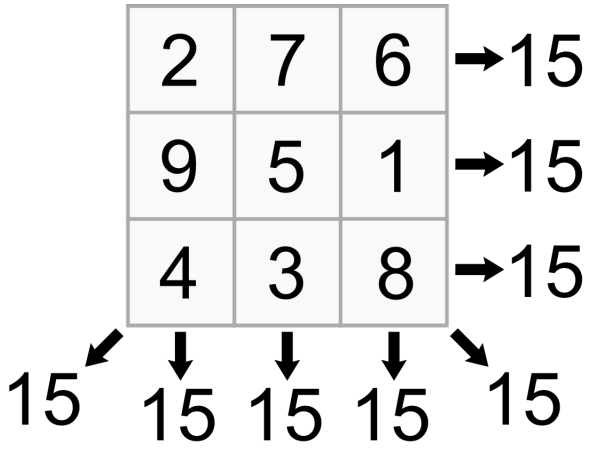

Figure 2.1: A magic square.

The second problem is to maximize a function of two variables. This function is given in Equation 2.1.

$$f(x,y) = \sin\left(\pi 10x + \frac{10}{1+y^2}\right) + \ln\left(x^2+y^2\right) \tag{2.1}$$

Although the same general algorithm can be utilized to solve both of these problems, certain functions within the algorithm will vary depending on the goal. Specifically, the fitness, crossover, and mutation functions will vary depending on the problem. The functions selected for each problem are described in the next sections.

Magic Square

Fitness Function

The magic square fitness function is a minimization function. It works by first calculating the "magic number". The magic number is a value that represents the target for each column, row, and diagonal sum. It is calculated according to Equation 2.2 where L is the length of one square dimension. As shown, the magic number is a function of L.

$$mn(L) = \frac{L + L^3}{2} \tag{2.2}$$

The fitness of the individual is calculated by summing the differences of the magic number and each row, column, and diagonal sum. This function is given in Equation 2.3 where s is a vector sum representing either a row, column, or diagonal sum and N is the number of vectors in a square.

$$F^{\alpha} = \sum_{i=0}^{N} mn - s_i \tag{2.3}$$

Individuals are considered more fit as their fitness value approaches 0. If an individual has a fitness of 0, a magic square has been found.

Crossover Function

As mentioned, the crossover function is a single-point crossover. Because each element in the square individual must be unique, we cannot simply

combine two parents to form valid children. For the crossover, a method described in the paper *Genetic Algorithm Solution of the TSP Avoiding Special Crossover and Mutation* [3] is utilized. Essentially, each parent individual is first translated into an inversion sequence. Next, these inversion sequences are crossed to generate two child inversion sequences. Finally, these child inversion sequences are translated back into actual square individuals.

Mutation Function

Again, because each element in the square individual must be unique, we cannot just mutate a child by updating one element with a random value. This would certainly cause a collision with one of the other square values. Instead, two random indexes are selected then swapped.

Function Maximization

This problem involves a different approach for fitness, crossover, and mutation.

Fitness Function

The fitness function here is simply the function that we are attempting to maximize. Because we are attempting to minimize the value in our genetic algorithm, we take the reciprocal of the function result as our fitness function. This means that a smaller value indicates better fitness. This allows us to use the same general approach for both problems.

Crossover Function

Because we are attempting to maximize a function of two variables, the individuals in the population have a length of two. This makes the crossover function rather simple. The first child's first element is the first parent's first element and the first child's second element is the second parent's second element. The second child's first element is the second parent's first element and the second child's second element is the first parent's second element.

Mutation Function

Since the length of each individual is only two, a random element is selected and replaced with a random number within the bounds specified in the

problem statement. In this project, the bounds are set at $3 \leq x \leq 10$ and $4 \leq y \leq 8$.

Experimentation

Analysis

In addition to the genetic algorithms, the brute-force algorithm for each problem was also tested for comparison. The results are given in the proceeding sections.

Magic Square

Generating magic squares was much more successful with the genetic algorithm than the brute-force approach. For experimentation, the genetic algorithm was first tested with different hyper-parameters for a 3x3 square. The results are given in Table 3.1 where the iterations column represents the number of iterations until convergence.

Generations	Population Size	Mutation Prob	Iterations
100	100	0.25	11
300	50	0.25	13
300	10	0.25	42

Table 4.1: Magic Square GA results.

Figure 3.1 shows the evolution of the mean population fitness for each generation. As shown, the fitness stagnates for some generations and improves dramatically for later generations.

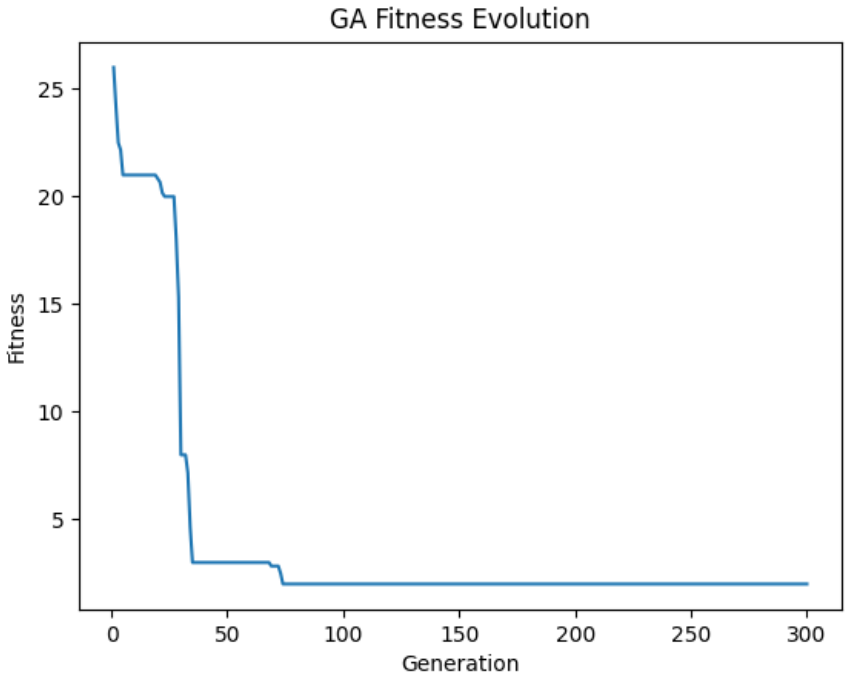

Figure 4.1: Magic square fitness evolution.

As for the brute-force implementation, it typically finds the solution anywhere from $20,000$ to $40,000$ iterations for a 3x3 square. This algorithm is slightly optimized in that it doesn't simply test every permutation of a possible square. It introduces a stochastic element by randomly swapping square values for each iteration. This works well for a 3x3 square, but generating squares larger than this is infeasible. With the genetic algorithm, 4x4 and 5x5 magic squares can be generated rather quickly. Any dimensions larger than this tend to prematurely converge.

Function Maximization

The genetic algorithm almost always finds the optimal solution for the function maximization problem in the first generation for populations greater than 5 individuals. Figure 3.2 shows the fitness evolution while using a population size of 5 over 300 generations.

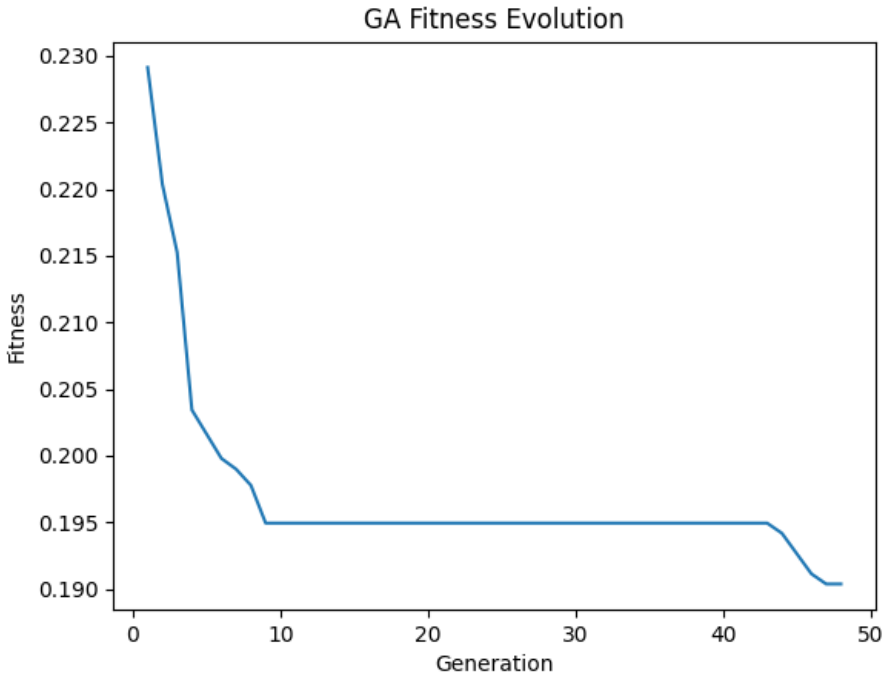

Figure 4.2: Function Maximization fitness evolution.

As shown in the figure, the GA converges after only 45 or so generations with a population size of 5. In contrast, the brute-force implementation finds the optimal values of x and y after 36 iterations. The optimal values for x and y are 10 and 4 respectively. These variables produce a result of 5.31.

Conclusion

The genetic algorithm is very efficient and applicable to many problems in the computer science domain. It is impressive that the same algorithm can be applied to widely different problems with good results. This was shown by solving the magic square and function maximization problems. One issue that crops up is premature convergence. This can be improved by modifying the hyper-parameters or even adding additional constraints to the model. One such constraint could be only allowing unique individuals in the population. This would ensure maximal diversity while maintaining those that are most

fit. One of the characteristics of this algorithm that makes it so robust is its level of configurability. Overall, this was a very interesting project exploring a powerful algorithm.

References

[1] X.-S. Yang, "Chapter 6 - Genetic Algorithms," in *Nature-Inspired Optimization Algorithms* (X.-S. Yang, ed.), pp. 91–100, Academic Press, second ed., 2021.

[2] S. Marsland, *Machine Learning An Algorithmic Perspective*. CRC Press, second ed., 2015.

[3] G. Üçoluk, "Genetic Algorithm Solution of the TSP Avoiding Special Crossover and Mutation," *Intelligent Automation and Soft Computing*, vol. 3, 01 2002.